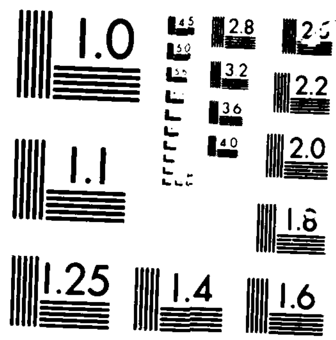
AD-A169 000 LOGISTICS WORKSTATION(U) COMPUTER AIDED PLANNING AND SCHEDULING INC ATLANTA GA APR 86 N00014-85-C-0815 1/1

UNCLASSIFIED F/G 5/1 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.8
3.2
3.6
4.0
4.5
5.0

CAPS

PHASE 1
FINAL REPORT
LOGISTICS WORKSTATION
CONTRACT NO. N00014-85-C-0815

April 1986

Computer Aided Planning
& Scheduling, Inc.

(12)

DTIC
ELECTE
MAY 1 2 1986
S D
D

Original contains color plates: All DTIC reproductions will be in black and white

PHASE 1
FINAL REPORT
LOGISTICS WORKSTATION
CONTRACT NO. N00014-85-C-0815

April 1986

Prepared By

COMPUTER AIDED PLANNING & SCHEDULING INC.
2900 Paces Ferry Road
Atlanta, Georgia 30339-3719

TABLE OF CONTENTS

I. INTRODUCTION

This research effort was motivated by the need for significantly better tools toaddress logistics planning problems and the opportunity to integrate the developing methodology in interactive optimization with the advances in microcomputer technology to meet this need. While there have been significant advances in the mathematics of optimization in logistics planning, they have not been accompanied by a corresponding increase in the use of optimization methodology by logistics planners. This is primarily due to three major factors:

(1) It is difficult to quantify many of the elements of logistics planning.

(2) Those elements of logistics planning which can be captured in optimization models often result in intractable models due to their size and complexity.

(3) Developing useful models for logistics planning is time consuming and frequently requires a level of expertise beyond that of the logistics planner.

The focus of this research is the design and construction of a logistics workstation with capability to significantly enhance analytical and decision making ability of logistics planners. This requires the integration of state-of-the-art data base technology, relevant optimization methodology, and human ingenuity in an environment which is both accessible and understandable to logistics planners.

Advances in microcomputers, both in computing power and color graphics capability, over the past five years has been phenomenal. They now have the power to quickly solve some of the most powerful mathematical models related to logistics planning (e.g., minimum cost network flow models, shortest path models, etc.). In addition, graphics capability allows display of model output in forms which are easily understood by the user. This, together with low cost and wide availability, makes the personal computer an ideal base to use for a workstation.

The most challenging part of the workstation design is the development of an environment which allows a logistics planner to *unify relevant* mathematical models and concepts in a framework which aids him in solving his specific problem. This requires more than simply making mathematical models available on personal computers. It requires an overall concept for addressing logistics planning problems and a structure which allows the user to easily develop a system customized to his particular problem.

Researchers have achieved little success in developing effective general logistics models. What has proven successful is modeling of various components of logistics planning problems with the human planner integrating the output of these models into an overall design (plan). This is generally a time consuming and cumbersome task requiring analytical sophistication beyond that frequently found in logistics planners.

Although mathematical models and concepts used to represent components of various logistics planning problems are frequently the same, both the problem context and the interfacing of these components vary widely. The definition of logistics as provided by the National Council of Logistics Management is:

> "the process of planning, implementing and controlling the efficient, cost effective flow and storage of raw materials, in-process inventory, finished goods, and related information from point of origin to point of consumption for the purpose of conforming to customer requirements."

While all of the problems under this definition are associated with material movement and storage, this includes a broad problem range. Rather than build a general workstation encompassing all of these problems, the design concept developed here permits the user to generate problem specific analysis macros (specially tailored procedures to accomplish complex logistics functions). These macros can either be used directly in the workstation or can be integrated into a customized interactive system for addressing the particular problem class of interest to the user.

The objectives of the Phase 1 effort were:

1. To develop modeling and control structures to accommodate design of an interactive logistics workstation.

2. To develop a prototype logistics workstation which encompasses basic vehicle routing and warehouse location problems.

3. To determine feasibility of developing logistics workstations for broad classes of logistics planning problems.

II. FAMILIES OF LOGISTICS PROBLEMS

Characteristics which make a logistics problem family particularly amenable to the workstation concept developed here are:

(1) there is a spatial representation of a significant component of each problem,

(2) significant elements of the problems are amenable to optimization or optimization based techniques,

(3) the problems are currently being solved by logistics planners using primarily their own insights and experience.

The following are problem families for which the workstation design provides a particularly attractive computing environment. Each of these problem families has generated a substantial published research literature related to optimization based models, algorithms, and concepts. However, most of this research has not been brought into a practical user environment. These problem families are also attractive for inclusion in a common workstation because there is a large overlap in both the optimization methods relevant to their solution and the graphics required to allow significant human interaction.

Routing

This problem family is primarily associated with movement of material from storage locations to points of use. The natural spatial representation of these problems is a geographical map. Relevant optimization models and algorithms include shortest path, minimum cost network flow (including assignment, transportation, and transshipment), generalized assignment, set covering, travelling salesman, branching, and backtracking.

The basic delivery problem is concerned with the assignment of loads to vehicles and the routing of the vehicles in moving material from a common warehouse to a number of different

customer locations. There are typically a variety of constraints on the vehicle loads, the routes, and the timing of deliveries.

A variation of the basic delivery problem is the pickup problem. Here, the material is picked up at a number of different locations and brought to a common location. The optimization methodology for addressing both the delivery and the pickup problems is very similar. With some extensions, this methodology can also be applied to problems with pickups and deliveries intermittently sequenced on the same routes.

A more complex class of problems are the dynamic pickup and delivery problems where demands for pickups and deliveries are scheduled and routed as they are received. These problems require elements of forecasting in addition to the assignment and routing components in the basic problems.

Scheduling

This problem family is concerned with the timing of events associated with material production, storage, and movement. The natural spatial representation is a bar chart. One of the principal analysis technique employed in addressing these problems is the methodology of network scheduling. Models in this family are described by time expanded networks. Paths through these time expanded networks provide the necessary scheduling information.

In a time expanded network, each original node is replaced by a copy of the node at a number of different points in time. The arcs in the time expanded network connect nodes at different times. The difference in the times of the nodes represents the traversal time of the original arc. Often the time expanded network can be preprocessed to reduce its size and complexity. For example, in a network assignment problem (e.g., deployment scheduling), where items (movement requirements) are assigned to processes (lift) over time, it is usually the case that the time expanded network is drastically reduced. The application of standard network optimization methods provide efficient solution techniques for these problems.

Another broad class of scheduling problems can be addressed by use of list schedules. These schedules are based on greedy procedures for creating ordered lists. While these methods are not usually optimum, in many cases bounds have been established for their worst case or average case performance.

Primarily, the network techniques apply when the activities can be split (i.e., processing the activity can be suspended for a time and then restarted). When activities cannot be split the list scheduling or more sophisticated implicit enumeration techniques are used.

Location

This problem family is concerned with the physical location and size of a variety of different kinds of facilities including production, storage, and service facilities. The primary cost tradeoff in these problems is the cost of building and running the facilities versus the transportation cost associated with getting material and services to and from the facilities. The natural spatial representation is again a geographical map. Relevant optimization models and techniques include: shortest path, assignment, generalized assignment, minimum cost flow, euclidean location, rectilinear location, set covering, branching, and backtracking.

There are frequently restrictions on the possible sizes of facilities. Generally this results from structural characteristics which make it practical to construct facilities only in discrete increments. There are also restrictions on the functions which can be put in particular facilities which result from both structural characteristics and from cost considerations. There may also be sourcing restrictions on facilities. One of the most common is a restriction that each customer be served by only a single facility. These constraints usually must be modeled by using discrete variables.

The potential locations may be restricted by a variety of factors including availability of labor, transportation, security, land, etc. The potential locations may already have facilities on them which further restrict the options with regard to space and functionality. Some of

these factors can frequently be modeled by using discrete variables. Other factors are very difficult to quantify. They can only be including in the planning process by knowledgeable human planners.

The workstation provides a framework for efficiently constructing procedures which take into account the human planner's knowledge together with discrete optimization methods which take advantage of the special mathematical structure of particular problems.

Layout

Layout problems involve a sizing component (determining the space required for each workstation or functional area, a location component (positioning the functional areas), and a network configuration component (determining the linking aisle or transportation structure). The natural spatial representation is a floor plan of the facility. Relevant optimization models include maximum spanning tree, minimum cut tree, rectilinear location, and euclidean location.

Layout problems almost always include elements which cannot be quantified. This together with their mathematical complexity and large associated cost make them ideal candidates for interactive optimization methodology.

The most successful layout approaches to date involve utilizing graphs as skeletons for the designs. The nodes represent functional areas and the links either represent the connecting aisle structure or the areas which should be adjacent in the layout. These skeletons can be obtained using the graph based optimization methods indicated above. Location models are useful in positioning the skeletons. The skeletons can be fleshed out using location/allocation models to allocate space to functional areas.

Much of the sizing of areas and the precise positioning of areas must be left to the human planner. The interactive graphics capability of the workstation design make this human intervention easy to accomplish.

Combination of Families

The richness of the logistics workstation allows the planner to utilize the basic workstation components to address very complex logistics planning problems made up of combinations of the above families. Areas which illustrate the power of the workstation in addressing complex modeling scenarios include military deployment, warehousing, and distribution.

The military deployment problems include both the static and dynamic delivery problems. Also, because of their size and the variety of different vehicle (asset) and cargo types, these problems typically require a hierarchal approach where different levels of aggregation are used at different levels. At the lower levels the ships, planes, and trucks must be scheduled and routed. However, at higher levels of the planning hierarchy, the capability of the transportation system to move material is considered in aggregate form. The workstation design has the capability to facilitate such a hierarchical approach. The planner would employ functions from routing and scheduling to address a deployment planning problem.

Many of the problems in warehousing are special cases of routing and location problems. The fundamental issues involve where and how to store material, how to pack and load the material, and how to schedule and sequence the storage and retrieval of material. The natural spatial representation is a floor plan of the warehouse. The storage configuration problems have a significant enough cost improvement and a planning leadtime long enough to justify the use of interactive approaches. Some of the packing problems (e.g., determining the pallet pattern for a product which is to be stored or shipped in large quantities) will also justify use of interactive methods. However, most of the storage and retrieval problems occur very frequently with not enough planning time or potential cost savings per occurrence to permit interactive methods.

For all three classes of problems, efficient optimization methods have been developed. Generally these methods are special cases of travelling salesman and location methods but are more tractable then the general problems because of the warehouse structure.

These methods will be included as systems functions in the workstations. Other methods including fast computerized heuristic methods as well as interactive methods can be constructed within the workstation design. As new specialized methods and algorithms are developed, they can be tested within the workstation. If they prove useful, they can be included either as macros or as systems functions.

Of major importance in logistics planning are the distribution (supply) functions. These involve the location and sizing of intermediate distribution centers, allocation of customers to these centers, and routing (transportation of material) from the centers. By combining the basic components of location, routing, and scheduling, the planner is able to manipulate families of complex distribution models and to synthesize configurations and operational strategies which efficient utilize limited resources. Configuration issues include how many distribution centers to open, where to locate them, sizing of them, and plant and customer allocation functions. Operational questions concern vehicle mix, routing, and scheduling to achieve maximum utilization of the centers and their assets.

III. CONCEPT

Three fundamental design concepts underlay the workstation. The first is that the workstation embodies the notion of interactive optimization. This implies that it is employs optimization as an extension of the human planner in solving problems rather than a replacement for the human. The second is that the workstation serves as either an analytical tool aiding a planner in solving a particular problem or a tool kit of macros used to construct an integrated system to aid in solving a class of problems. The third is that the workstation is continually evolving; new systems functions and user defined macros can be added to the system as they are required.

Interactive Optimization

Interactive optimization is a problem solving methodology which embodies optimization components in a flexible structure with significant human participation and control. When properly implemented, an interactive optimization system has the human planner and the optimization components working together as a team. The logistics planner is team leader and decision maker while the computer and optimization components do much of the work providing information and advice to the planner. As with any effective team, it is essential that the human and computer components be able to communicate with each other.

The communication concept employed in the workstation is that human effort required be as little as possible. This allows the planner to focus attention on the problem to be solved rather than on complexities of running the system. This also facilitates training of users of the system.

Two characteristics of the workstation which enhance communication are

(1) the system is menu driven, and

(2) communication is in a language which is as natural as possible for the logistics planner.

For the workstation, the user is provided with a menu structure of available options. The desired option is selected by pressing a single key on the computer keyboard. Graphics displays are used when possible to reduce the amount of human effort required for data input. For example, to input data such as potential sites for facilities, the user can indicate the locations on a geographical display by positioning the cursor over the location and pressing a single key. Other specific communication characteristics of the workstation will be discussed in the next section.

Workstation Modes

In order to provide maximum problem solving capability, four different modes have been designed into the workstation. These modes are called macro mode, analysis mode, production mode, and configuration mode.

In macro mode, the user is able to create macros containing a particular sequence of system functions. (When a macro is executed, systems functions in it are executed in the order specified.) Macro mode is essentially an editor wherein the user builds and changes macros.

In analysis mode, the user is able to execute system functions and user defined macros in any order desired. The workstation control structures provide menus to aid in determining functions and macros available and in specifying inputs and outputs for the functions. Analysis mode is primarily used in the research and development phase of building a production mode system. For example, analysis mode might be used in testing different sequences of system functions to determine the best configuration for a particular daily delivery routing system. It might also be used for a one time analysis such as determining the best location for a major facility.

In production mode, the user is restricted to the functionality defined by the controlling macros. Production mode is primarily used for fairly refined systems which are used repetitively. An example of production mode is a set of macros constituting an interactive dispatching system for daily delivery.

In configuration mode the user is able to specify a number of controlling parameters for the workstation including the menu items for analysis mode and the functions attached to these menu items. When specifying menus in configuration mode or inside user defined macros, the menu terminology is defined appropriately for the particular application. System function definitions and menus provided by the workstation, in macro and analysis modes, use terms such as "travelling salesman", "2-opt", "euclidean location", etc. These are taken, whenever possible, from terms in common use (at least in the more analytical segment of the logistics community).

System Evolution

Since it is not possible to anticipate all of the system functions required for logistics planning, the workstation has a library of systems functions which can be added to as necessary. Modular design also permits addition of new menus and submenus. The workstation is coded in standard "C" language for the IBM PC and utilizes the Virtual Device Interface (VDI) graphics standard for all graphics. This offers maximum flexibility in evolving the workstation as new needs arise and enhancements to personal computers are developed.

IV. FUNCTIONAL DESCRIPTION

In order to use the workstation, the first step is to boot up the personal computer and initiate the workstation program. System functions are executed by typing a "/", which displays the system menus, and pressing the appropriate function key or moving the cursor to the appropriate menu item and pressing "Enter". The logistics workstation has a reserved function key for "help." The planner can obtain help from anywhere in the system. The help function provides information related to the use of the workstation. A special key, the escape key, is provided to pause and discontinue functions within the workstation.

Master Menu

The Master menu is the basic entry point for the logistics workstation. From this menu, the planner can access the other menus which provide the functionality of the logistics workstation. Master menu items are:

1. Production: This puts the workstation into production mode, allowing execution of various predefined systems of macros.

2. Analysis: This puts the workstation into analysis mode, permitting execution of system functions or user defined macros.

3. Macro: This puts the workstation into macro mode, allowing construction and editing of macros as well as specification of a menu structure and associated system functions and macros.

4. Configuration: This puts the workstation into configuration mode, accommodating changes in basic parameters such as menu items in analysis mode and functions assigned to them.

5. Exit: This exits the workstation.

Production

Submenus under the production menu allow the user to find, examine, or execute a production macro. Submenu items include:

1. Load: This allows loading of a production macro into memory.

2. Run: This executes a selected production macro.

3. List: This lists commands in a selected production macro.

4. Directory: This gives a directory of available production run macros with a brief description of each.

5. Exit: This returns the planner to the master menu.

A production macro can be selected by typing in the macro name or by using the left and right cursor keys to find the desired macro and then pressing "Enter".

Analysis

Submenus under the analysis menu permit the user to execute system functions or macros previously constructed in macro mode. Functions accessed within each menu element can be specified in configuration mode. The major menu in analysis mode has the following elements.

1. Files: The set of system functions which manipulate data files. This includes reading, writing, editing, merging, partitioning, sorting, and aggregating data files. Rather than have its own file editor, the workstation will allow any of a number of commercially available editors to be called within this menu.

2. Display: The set of system functions which provide graphics displays. This includes plotting points, drawing routes, coloring point clusters, zooming areas of the screen, drawing background maps, and drawing time distance charts.

3. Reports: Generation of customized reports from stored data files or files in memory. Examples include route summary reports, truck load reports, and customer assignment reports.

4. Compute: All of the optimization and computation functions such as: determine

spacefilling curve, generate travelling salesman tour, locate skeleton, assign customers to routes, and calculate route costs.

5. Macros: All of the user defined macros. Whenever a particular sequence of systems functions are to be called repetitively, they can be put in a macro. An example is a procedure for generating truck loads and routes which (a) generates a spacefilling curve, (b) develops a travelling salesman (TSP) tour based on the space filling curve, and (c) breaks the TSP tour into segments (amounts) which can be loaded onto a single truck. Since each of (a), (b), and (c) are systems functions, they could be executed one at a time. However, by creating a macro containing these three functions, they can be executed as if they were a single function.

6. Exit: This returns the planner to the master menu.

Macro

Macro mode permits the user to define a macro consisting of a specific sequence of systems functions and previously defined macros. Macros can be command macros which can be series of system functions or other command macros, or menu macros which define the options of a menu. When a macro is executed in production or analysis mode, all of the systems functions and macros within the macro are automatically executed. Macro mode also allows macros to be attached to menus. This is particularly useful when structuring a system to run in production mode. The major menu items in macro mode are the following.

1. Command: To construct a command macro, the planner can either types names of system functions or other macros into a macro file or access systems functions in macro mode in exactly the same way they are accessed in analysis mode. Instead of being executed, the systems functions are placed in a macro file. In effect, the planner directs the system to "remember" a set of keystrokes for later use. A commercial editor can also be used to construct a command macro.

2. Menu: Construction of a special menu macro allows the definition of up to seven menu items per screen. Associated with each menu item is a one word description which is displayed inside the menu block, a one line description which is displayed whenever the cursor is positioned on the menu block, and a single system function or user defined macro name which is executed when the menu item is accessed. Since menus are themselves macros, a menu can be executed from another menu. This allows construction of complex systems which are completely menu driven.

3. Edit: To edit a macro the system calls one of a number of commercially available editors. This allows the system to take advantage of excellent existing editors. It also allows the system to incorporate improvements in editors as they become available.

4. Directory: The directory is used to find and review macros in the current workstation library.

5. Run: When building and editing macros, it is frequently necessary to run the macro to determine its correctness.

6. Exit: This returns to the master menu which allows entry into other workstation modes.

Macros are interpreted at execution time after which the appropriate functions are executed.

Configuration

Configuration mode is used to modify parameters associated with analysis mode. As the workstation evolves, the number of systems functions and user defined macros increase. The user would not normally need all of the available functions. Since both computer memory and planner time are dependent on the number of functions, it is desirable to have a mechanism for limiting functions loaded into the system to those of value to the particular user. The major menu items in configuration mode are the following.

1. Files: This allows specification of the set of data file manipulation functions which are available in analysis mode.

2. Display: This allows specification of the set of graphics display system functions which are available in analysis mode.

3. Reports: This allows specification of the set of customized reporting functions which are available in analysis mode.

4. Compute: This allows specification of the set of optimization and computation functions which are available in analysis mode.

5. Macros: This allows specification of the set of user defined macros which are available in analysis mode.

6. Exit: This returns the planner to the master menu.

Limitations

Since the workstation is being designed for microcomputers, its major limitation is with respect to computing power. However, microcomputers continue to increase in power at an astonishing rate. It seems likely that the speed, memory, and storage capacity of these computers will soon be sufficient for all but the largest logistics applications.

V. PROPOSED FUNCTIONALITY

The workstation will require a rich library of analysis functions to support the broad class of logistics problems indicated above. Functions listed below provide the basis for analysis in each of the families of problems discussed above. The function library is partitioned into three general areas (display, files, and optimization) plus areas which are specific to each problem family (routing, scheduling, location, and layout).

Display

There are a number of functions which manipulate the display screen in the workstation. Some of these are described below.

FUNCTION: **Redraw Current Active Graphics Screen**

INPUTS: None

OUTPUTS: None

DESCRIPTION: This command will redraw the active area to fill the entire graphics screen.

FUNCTION: **Identify a Point, Route, or Block**

INPUTS: A point and the type of structure containing the point.

OUTPUTS: Information about the point or structure containing the point.

DESCRIPTION: This command allows the identification of a point, route or block. After selecting this command, a point and the type of containing structure is specified. The system will display the structure identification (e.g., type, weight, volume, and time windows for the point and route number, length, vehicle type, weight, and volume if the containing structure is a route).

FUNCTION: **Grid Aggregate Weight and Volume**

INPUTS: None

OUTPUTS: Pickup and delivery weight and volume for each grid area.

DESCRIPTION: This command imposes a rectangular grid over the graphics display area. The total unrouted demand weight and volume within each grid area is displayed numerically inside that grid area.

FUNCTION: **Draw Routes**

INPUTS: Route number.

OUTPUTS: Single or multiple routes displayed on the screen.

DESCRIPTION: This command displays selected routes.

FUNCTION: **Draw Network**

INPUTS: Nodes, node coordinates, and arcs.

OUTPUTS: Network displayed on the screen.

DESCRIPTION: This command draws a circle on the screen for each node and for each arc draws a straight lines connecting the corresponding nodes.

FUNCTION: **Draw Points**

INPUTS: Point type, coordinates, and color.

OUTPUTS: Single or multiple points displayed on the screen.

DESCRIPTION: This command plots points on the screen at the coordinates specified. The type designator allows display of sets of points with the same characteristics (e.g., all points corresponding to customers assigned to a particular facility). It also allows for different shapes corresponding to different point types (e.g., squares represent warehouses and circles represent customers).

FUNCTION: **Draw (Layout) Blocks**

INPUTS: Facility and block number.

OUTPUTS: Single or multiple blocks displayed on the screen.

DESCRIPTION: This command displays selected layout blocks (areas of facilities).

FUNCTION: **Draw (Scheduling) Blocks**

INPUTS: Activity name, time window, scheduled start time, periods scheduled, and level of effort in each period.

OUTPUTS: Single or multiple activity bar chart displayed on the screen.

DESCRIPTION: This command displays selected scheduling blocks (activities to be scheduled) positioned on a time scale. It also displays start and end of desired scheduling window.

FUNCTION: **Draw Route Skeletons**

INPUTS: Route numbers.

OUTPUTS: Selected route skeletons are drawn on the current display.

DESCRIPTION: This command draws selected route skeletons.

FUNCTION: **Draw Facility Skeletons**

INPUTS: Facility numbers.

OUTPUTS: Selected facility skeletons are drawn on the current display.

DESCRIPTION: This command draws selected facility skeletons.

FUNCTION: **Toggle Route Stems On/Off**

INPUTS: None

OUTPUTS: Current display redrawn with/without route stems.

DESCRIPTION: This command turns on/off route stems (the route legs between the route and the facility)

FUNCTION: **Draw Background Map**

INPUTS: None

OUTPUTS: Current display redrawn with background map.

DESCRIPTION: This command turns on the background map. A background map is useful for orientation within a set of points; however, background maps can contribute to clutter of the display and should sometimes be omitted from the screen.

FUNCTION: **Print to Printer**

INPUTS: None

OUTPUTS: Current display is printed on a printer.

DESCRIPTION: This command prints a copy of the current graphics screen on the printer.

FUNCTION: **Draw Icon**

INPUTS: Icon name and <.> key.

OUTPUTS: Icon will be added to the screen.

DESCRIPTION: When this command is selected, the system prompts for an icon. The planner is prompted to provide a location on the display for the icon to be drawn. He uses the arrow keys to locate the icon, press the <.> key, after which the icon is drawn.

FUNCTION: **Draw Time-Distance Chart**

INPUTS: Route number.

OUTPUTS: Time-distance chart displayed on the screen.

DESCRIPTION: This function is used to draw a time-distance chart for any route.

FUNCTION:	**Zoom an Area**
INPUTS:	The two corner points of the area to zoom.
OUTPUTS:	The zoom area will be redrawn to fill the screen.
DESCRIPTION:	This command is used to define a portion of the graphics screen for closer examination.

FUNCTION:	**Restore (Unzoom) the Original Display Window**
INPUTS:	None
OUTPUTS:	The original problem area is redrawn on the screen.
DESCRIPTION:	This command restores the complete problem space to the original active display window (no matter how many zoom areas have been drawn). All points in the original problem are redrawn to the full screen.

Files

Files are the basic input/output structures of most system functions. For this reason the workstation will have extensive utilities which allow manipulation of files. Because of the wide diversity of required inputs to the system functions it is not practical to fix the format of files. Rather, each input or output data file will have an associated field map file, which simply defines where variables may be found in the data file. Field map files are fully explained in **Chapter VII: Implementation**. Functions which manipulate workstation files and field map files are described below.

FUNCTION:	**Read a File**
INPUTS:	File name.
OUTPUTS:	None
DESCRIPTION:	This command allows the planner to select a file and bring it into the workspace.

FUNCTION: **Save a File**

INPUTS: File name.

OUTPUTS: File saved to disk.

DESCRIPTION: This command allows the planner to select a file currently in the workspace and save it to disk.

FUNCTION: **Look at a File**

INPUTS: File name.

OUTPUTS: First few lines of the file.

DESCRIPTION: This command allows the planner to select a file and see the first few lines of it. This is useful in determining whether this is the proper file for some use.

FUNCTION: **Discard a File**

INPUTS: File name.

OUTPUTS: None

DESCRIPTION: This command allows the planner to delete a file. This helps manage disk space.

FUNCTION: **Edit a File**

INPUTS: File name.

OUTPUTS: The planner is switched to the editor selected during configuration mode.

DESCRIPTION: This command allows the planner to edit a file. When the command is evoked, the system chains to the editor provided to it during configuration mode, and the planner is permitted to make additions, deletions, and changes to it.

FUNCTION: **Merge Files**

INPUTS: File names.

OUTPUTS: Merged files.

DESCRIPTION: This command allows the planner to combine a set of files. Each file is appended to the previous one supplied.

FUNCTION: **Partition a File**

INPUTS: File name, row or column number(s), and new file names.

OUTPUTS: Partitioned files.

DESCRIPTION: This command allows the planner to breakup a file. The selected file is partitioned into a set of files, one new one for each row or column number supplied.

FUNCTION: **Sort a File**

INPUTS: File name and field descriptors.

OUTPUTS: Sorted file.

DESCRIPTION: This command allows the planner to sort a file. The lines of the file are sorted with respect to the data in the fields provided.

FUNCTION: **Aggregate a File**

INPUTS: Aggregation method and percentage level.

OUTPUTS: File containing aggregated data.

DESCRIPTION: This command will provide the user with a wide range of aggregation techniques. On selecting this command, the planner is prompted to select an aggregation method and level. The system applies the selected aggregation method (distance, volume, weight, etc.) to achieve an aggregation of the points inside the current display window to the desired aggregation level (percent reduction in number of points). When the aggregation is complete, the screen is erased and the aggregated points are drawn. If the planner presses the <Esc> key while the aggregation process is underway, the system comes to a orderly stop and the aggregation results, to this point, are displayed.

Optimization

The workstation will support a variety of high-level procedures for performing various optimization functions. They serve as the strength of the workstation to aid the planner in addressing complex logistics problems. Some of the optimization procedures to be included are listed below.

FUNCTION: **Linear Programming**

INPUTS: Coefficient matrix, right-hand-side vector, cost vector.

OUTPUTS: Primal and dual variables.

DESCRIPTION: This function accepts a linear program, applies the simplex algorithm, and returns the primal and dual variable values and the associated cost. If the **<Esc>** key is pressed during execution of the procedure, the algorithm will find an orderly stopping point and return the solution to this point.

FUNCTION: **Shortest Path**

INPUTS: Nodes and arcs.

OUTPUTS: Path.

DESCRIPTION: This function accepts a node and an arc file, applies a shortest path algorithm, and returns a file of the arcs in the shortest path. If the **<Esc>** key is pressed during execution of the procedure, the algorithm will find an orderly stopping point and return the solution to this point.

FUNCTION: **Minimal Cost Flow**

INPUTS: Nodes and arcs.

OUTPUTS: Flow values and dual variable values.

DESCRIPTION: This function accepts a node and an arc file, applies a minimal cost network flow algorithm, and returns a file of the flows on the arcs in the solution and a file of the dual variables for the nodes at optimality. If the **<Esc>** key is pressed during execution of the procedure, the algorithm will find an orderly stopping point and return the solution to this point.

FUNCTION: **Generalized Minimal Cost Flow**

INPUTS: Nodes and arcs.

OUTPUTS: Flow values and dual variables.

DESCRIPTION: This function accepts a node and an arc file, applies a generalized minimal cost network flow algorithm, and returns a file of the flows on the arcs in the solution and a file of the dual variables for the nodes at optimality. If the <Esc> key is pressed during execution of the procedure, the algorithm will find an orderly stopping point and return the solution to this point.

FUNCTION: **Minimal Spanning Tree**

INPUTS: Nodes and arcs.

OUTPUTS: Arcs.

DESCRIPTION: This function accepts a node and an arc file, applies a minimal spanning tree algorithm, and returns a file of the arcs in the minimal spanning tree. If the <Esc> key is pressed during execution of the procedure, the algorithm will find an orderly stopping point and return the solution to this point.

FUNCTION: **Longest Path**

INPUTS: Nodes and arcs.

OUTPUTS: Path.

DESCRIPTION: This function accepts a node and an arc file, applies a longest path algorithm, and returns a file of the arcs in the shortest path. If the **<Esc>** key is pressed during execution of the procedure, the algorithm will find an orderly stopping point and return the solution to this point.

FUNCTION: **2-Opt Tour**

INPUTS: Nodes.

OUTPUTS: Nodes.

DESCRIPTION: This function accepts a node file, applies a 2-opt interchange algorithm, and returns a file of the nodes in order in the improved path (tour). If the **<Esc>** key is pressed during execution of the procedure, the algorithm will find an orderly stopping point and return the solution to this point.

FUNCTION: **3-Opt Tour**

INPUTS: Nodes.

OUTPUTS: Nodes.

DESCRIPTION: This function accepts a node file, applies a 3-opt interchange algorithm, and returns a file of the nodes in order in the improved path (tour). If the **<Esc>** key is pressed during execution of the procedure, the algorithm will find an orderly stopping point and return the solution to this point.

FUNCTION: **Spacefilling Curve Tour**

INPUTS: Nodes.

OUTPUTS: Nodes.

DESCRIPTION: This function accepts a node file, applies a spacefilling algorithm, and returns a file of the nodes in order in the spacefilling tour. If the **<Esc>** key is pressed during execution of the procedure, the algorithm will find an orderly stopping point and return the solution to this point.

FUNCTION: **Time Expanding a Network**

INPUTS: Nodes and arcs in the network, nodes to be expanded, time increment, time period for each node.

OUTPUTS: Nodes and arcs in the time expanded network.

DESCRIPTION: This function accepts a network and returns a time expanded network.

FUNCTION: **Minimum Cut Tree**

INPUTS: Nodes and arcs in the network and arc capacities.

OUTPUTS: Nodes and arcs in the cut tree.

DESCRIPTION: This function accepts a network and returns a minimum cut tree for the network.

FUNCTION: **Set Covering**

INPUTS: Zero and one coefficient matrix and cost vector.

OUTPUTS: Variables in the cover.

DESCRIPTION: This function accepts a set covering problem and applies a cost splitting algorithm to generate a cover.

FUNCTION: **Implicit Enumeration**

INPUTS: Variable to be fixed and vector of search to date.

OUTPUTS: Vector of search to date and next restricted problem to be solved.

DESCRIPTION: This function handles the bookkeeping for a depth first tree search over a set of zero-one variables.

FUNCTION: **Euclidean Location**

INPUTS: Points and associated weights.

OUTPUTS: Coordinates of location.

DESCRIPTION: This function accepts a set of points and their associated weights and finds the optimum location for a facility to serve the set of points assuming weighted euclidean distance as the objective.

FUNCTION: **Rectilinear Location**

INPUTS: Points and associated weights.

OUTPUTS: Coordinates of location.

DESCRIPTION: This function accepts a set of points and their associated weights and finds the optimum location for a facility to serve the set of points assuming weighted rectilinear distance as the objective.

Vehicle Routing

Vehicle routing functionality will be a fundamental part of the workstation. The workstation will use state-of-the-art optimization techniques in solve vehicle routing problems. These techniques use the concept of route skeletons. A route skeleton is a geometric abstraction of a route. A skeleton may be a simple shape, such as a point or a line, or a more complex shape, such as an ellipse. The routing algorithms work with route skeletons rather than the actual routes, greatly improving solution efficiency. Some functions associated with vehicle routing in the workstation are listed below.

FUNCTION: **Add a Route Skeleton**

INPUTS: Route, depot, type, and position of the route skeleton.

OUTPUTS: The route skeleton is drawn on the current display.

DESCRIPTION: This command creates a route skeleton and prompts the planner to place it on the graphics screen. When he is prompted for the route skeleton type, he must press a function key to indicate his choice:

- Free Line Segment
- Fixed Line Segment
- Point

The system then prompts for the necessary parameters to specify the route skeleton. For example, the system will prompt for a depot if multiple depots exist, and a vehicle type if multiple vehicle types exist.

FUNCTION: **Delete a Route Skeleton**

INPUTS: Route skeleton number.

OUTPUTS: None

DESCRIPTION: When this option is selected, all points in the route are freed and the route skeleton is deleted from the current solution. The route will be erased from the screen.

FUNCTION: **Alter a Route Skeleton**

INPUTS: Depot, type, and point(s) defining the new route skeleton.

OUTPUTS: The new route skeleton is drawn in the current display.

DESCRIPTION: This command allows the planner to alter the basic form (Line, or Point) of a route skeleton, as well as its location and associated depot and vehicle. The system will erase the selected route skeleton and prompt for a new one. All points assigned to the old route skeleton will be automatically assigned to the new one. If there is more than one depot then the system will prompt for a depot for the route skeleton. If there is more than one vehicle type then the system will prompt for a vehicle type for the route skeleton.

FUNCTION: **Position a Route Skeleton with Respect to a Set of Points**

INPUTS: Set of points and route skeletons.

OUTPUTS: The positioned route skeleton is drawn on the current display.

DESCRIPTION: With this command, the selected route skeleton is automatically adjusted to better reflect the shape of the route it represents. However, its basic form (Line or Point) does not change.

FUNCTION: **Breakup Route into Route Skeletons**

INPUTS: A route and a set of separation criteria.

OUTPUTS: Route skeletons.

DESCRIPTION: With this command, the selected route skeleton is automatically located to reflect the shape of the portion of the great route it represents.

FUNCTION: **Insert a Point Into a Route**

INPUTS: The unrouted point to be inserted and the routed point after which the unrouted point is inserted.

OUTPUTS: The route is redrawn to include the point.

DESCRIPTION: With this command, an unrouted point is inserted into a route <u>after</u> a routed point. If the merged route violates a route constraint, the planner is prompted to confirm this is acceptable.

FUNCTION: **Delete a Point From a Route**

INPUTS: Point to be deleted.

OUTPUTS: Route redrawn without specified point.

DESCRIPTION: With this command, a routed point is deleted from the route to which it is currently assigned. Once this command is selected, then the point to be deleted is specified. The system will free the demand point, and adjust the route.

FUNCTION: **Merge Two Routes**

INPUTS: A point on each route.

OUTPUTS: The two routes are erased and the combined route is redrawn.

DESCRIPTION: With this command, the Points on two routes are combined to form one route. The route skeleton of the first route is taken to be the route skeleton representing the combined routes. The planner must select this command, specify the first route, then specify the second route. The system will combine both routes, resequence the new route, erase both routes and draw a combined route. If the merged route violates a route constraint, the planner is prompted to confirm if this is acceptable.

FUNCTION: **Separate a Route Into Two Routes**

INPUTS: The point after which the route is to be separated.

OUTPUTS: The single route is erased and the separate routes are redrawn.

DESCRIPTION: To separate a route into two routes, the planner must first select this command, then specify the last point to be included in the current route. The system will create a new route made up of the remaining points in the route. The system will make two copies of the original route skeleton - one for each of the new routes.

FUNCTION: **Clear All The Points in a Route**

INPUTS: A point on the route.

OUTPUTS: All points on the route are freed from the route and the route is deleted.

DESCRIPTION: When the planner selects this command, the points assigned to a current route are cleared (re-specified as unrouted). This command is selected and the planner specifies the route to be cleared. The route is erased but the route skeleton representing the route remains.

FUNCTION: **Add Points to Routes**

INPUTS: Number of points to be added and selection criteria.

OUTPUTS: As points are added, the routes are redrawn.

DESCRIPTION: This command sequentially adds points to the current routes. On selecting this command the planner is prompted for the number of points to be added. The system will then sequentially add points to the current solution until the specified number of points have been added, or until no further additions are possible with the current set of route skeletons. An entry of **0** (zero) will cause no points to be added.

Scheduling

Scheduling can be considered as a variation of routing where the routing is being performed in time rather than distance. Hence, many of the logical scheduling functions are very similar to their counterpart in routing. The scheduling functions allow schedules to be manually manipulated as well as allowing the modification and display of schedules generated by optimization models.

FUNCTION: **Insert an Activity Into a Schedule**

INPUTS: The unscheduled activity to be inserted and the scheduled activity after which the unscheduled activity is to be inserted.

OUTPUTS: The schedule is adjusted to include the activity.

DESCRIPTION: With this command, an unscheduled activity is inserted into a schedule **after** a scheduled activity. If the merged schedule violates a schedule constraint, the planner is prompted to confirm this is acceptable.

FUNCTION: **Delete an Activity From a Schedule**

INPUTS: Activity to be deleted.

OUTPUTS: Schedule redrawn without specified activity.

DESCRIPTION: With this command, a scheduled activity is deleted from the schedule to which it is currently assigned. Once this command is selected, then the activity to be deleted is specified. The system will free the demand activity, and adjust the schedule.

FUNCTION: **Merge Two Schedules**

INPUTS: A activity on each schedule.

OUTPUTS: The two schedules are erased and the combined schedule is redrawn.

DESCRIPTION: With this command, the activities on two schedules are combined to form one schedule. The planner must select this command, specify the first schedule, then specify the second schedule. The system will combine both schedules, resequence the new schedule, erase both schedules and draw a combined schedule. If the merged schedule violates a schedule constraint, the planner is prompted to confirm if this is acceptable.

FUNCTION: **Separate a Schedule Into Two Schedules**

INPUTS: The activity after which the schedule is to be separated.

OUTPUTS: The single schedule is erased and the separate schedules are redrawn.

DESCRIPTION: To separate a schedule into two schedules, the planner must first select this command, then specify the last activity to be included in the current schedule. The system will create a new schedule made up of the remaining activities in the schedule.

FUNCTION: **Clear All the Activities in a Schedule**

INPUTS: A activity on the schedule.

OUTPUTS: All activities on the schedule are freed from the schedule and the schedule is deleted.

DESCRIPTION: When the planner selects this command, the activities assigned to a current schedule are cleared (re-specified as unscheduled). This command is selected and the planner specifies the schedule to be cleared.

FUNCTION: **Add Activities to Schedules**

INPUTS: Number of activities to be added and selection criteria.

OUTPUTS: As activities are added, the schedules are redrawn.

DESCRIPTION: This command sequentially adds activities to the current schedules. On selecting this command the planner is prompted for the number of activities to be added. The system will then sequentially add activities to the current solution until the specified number of activities have been added, or until no further additions are possible. An entry of **0** (zero) will cause no activities to be added.

FUNCTION: **Generate List Schedule**

INPUTS: Activities and criteria for ordering.

OUTPUTS: Activity schedule.

DESCRIPTION: This command generates a schedule of activities based on an ordered list. The list can be ordered on a variety of standard criteria including processing time, due date, slack time, lateness, and combinations of these criteria.

Location

In addition to the functions described earlier which support general manipulation of points and files, there are a number of specific system functions providing capability in location modeling and analysis. Like the vehicle routing system functions, the location system functions make use of skeletons, which succinctly capture spatial information. Some of these are listed below.

FUNCTION: **Add a Facility Skeleton**

INPUTS: Depot (facility), type, and position of the facility skeleton.

OUTPUTS: The facility skeleton is drawn on the current display.

DESCRIPTION: This command creates a facility skeleton and prompts the planner to place it on the graphics screen. When he is prompted for the facility skeleton type, he must press a function key to indicate his choice:

- Line Segments (star)
- Point

The system then prompts for the necessary parameters to specify the facility skeleton. For example, the system will prompt for a depot if multiple depots exist, and a vehicle type if multiple vehicle types exist.

FUNCTION: **Delete a Facility Skeleton**

INPUTS: Skeleton.

OUTPUTS: None

DESCRIPTION: When this option is selected, all points assigned to the facility are freed and the facility skeleton is deleted from the current solution.

FUNCTION: **Alter a Facility Skeleton**

INPUTS: Depot, type, and point(s) defining the new facility skeleton.

OUTPUTS: The new facility skeleton is drawn in the current display.

DESCRIPTION: This command allows the planner to alter the basic form (Line, or Point) of a facility skeleton, as well as its location and associated depot and vehicle. The system will erase the selected facility skeleton and prompt for a new one. All points assigned to the old facility skeleton will be automatically assigned to the new one. If there is more than one depot then the system will prompt for a depot for the facility skeleton. If there is more than one vehicle type then the system will prompt for a vehicle type for the facility skeleton.

FUNCTION: **Position a Facility Skeleton on a Set of Points**

INPUTS: A set of points and a skeleton type.

OUTPUTS: The current facility skeleton is erased and the repositioned facility skeleton is drawn on the current display.

DESCRIPTION: With this command, the selected facility skeleton is automatically adjusted to better reflect the shape of the customer assignment it represents. The system automatically reshapes and relocates the facility skeleton based on the points currently assigned to the corresponding facility.

FUNCTION: **Breakup Route into Facility Skeletons**

INPUTS: A route and a set of separation criteria.

OUTPUTS: Facility skeletons.

DESCRIPTION: With this command, the selected facility skeleton is automatically located to reflect the shape of the portion of the great route it represents.

FUNCTION: **Insert a Point Into a Facility Assignment**

INPUTS: The unassigned point to be inserted.

OUTPUTS: The facility assignment is redrawn to include the point.

DESCRIPTION: With this command, an unassigned point is inserted into a facility assignment. If the merged assignment violates a facility constraint, the planner is prompted to confirm this is acceptable.

FUNCTION: **Delete a Point From a Facility Assignment**

INPUTS: Point to be deleted.

OUTPUTS: Facility assignment redrawn without specified point.

DESCRIPTION: With this command, an assigned point is deleted from the facility assignment to which it is currently assigned. Once this command is selected, then the point to be deleted is specified. The system will free the demand point.

FUNCTION: **Merge Two Facilities**

INPUTS: Two facilities.

OUTPUTS: The two facility assignments are erased and the combined facility assignment is redrawn.

DESCRIPTION: With this command, the Points in two facility assignments are combined to form one assignment. The facility skeleton of the first assignment is taken to be the facility skeleton representing the combined facility. The planner must select this command, specify the first facility, then specify the second facility. The system will combine both facilities and draw a combined facility assignment. If the merged facility assignment violates a facility constraint, the planner is prompted to confirm if this is acceptable.

FUNCTION: **Separate a Facility Into Two Facilities**

INPUTS: The points to be separated.

OUTPUTS: The points to be separated are erased and the separate facility assignments are redrawn.

DESCRIPTION: To separate a facility into two facilities the planner must first select this command, then specify the each point to be included in the new facility. This is accomplished by answering a yes/no question which the system presents for each point in the current facility assignment. Before each question is asked, the system places the cursor on the particular point. The system will create a new facility made up of the indicated points. The system will make two copies of the original facility skeleton - one for each of the new facilities.

FUNCTION: **Clear All The Points in a Facility Assignment**

INPUTS: The facility.

OUTPUTS: All points in the facility assignment are freed.

DESCRIPTION: When the planner selects this command, the points assigned to a current facility are cleared (re-specified as unassigned). This command is selected and the planner specifies the facility to be cleared. The facility assignment is erased but the skeleton representing the facility remains.

FUNCTION: **Add Points to Facilities**

INPUTS: Number of points to be added.

OUTPUTS: As points are added, the facility assignments are redrawn.

DESCRIPTION: This command sequentially adds points to the current facilities. On selecting this command the planner is prompted for the number of points to be added. The system will then sequentially add points to the current solution until the specified number of points have been added, or until no further additions are possible with the current set of facility skeletons. An entry of **0** (zero) will cause no points to be added.

Layout

Layout problems are concerned with positioning areas while location problems are concerned with positioning points. Hence, many of the layout functions are analogous to layout functions but additional functions are required to handle the additional dimension of area.

FUNCTION: **Generate a Layout Skeleton**

INPUTS: Skeleton type and area to area travel matrix.

OUTPUTS: The layout skeleton is drawn on the current display.

DESCRIPTION: This function executes an optimization model which in turn specifies a layout skeleton and prompts the planner to position it on the graphics screen. When he is prompted for the layout skeleton type, he must press a function key to indicate his choice:

- Adjacency graph
- Aisle graph

The system then prompts for the necessary parameters to specify the layout skeleton. For example, the system will prompt for the optimization model to be used to generate a skeleton of the type specified.

FUNCTION: **Delete a Layout Skeleton**

INPUTS: Skeleton.

OUTPUTS: None

DESCRIPTION: When this option is selected, all areas assigned to the layout are freed and the layout skeleton is deleted from the current solution.

FUNCTION: **Insert a Area Into a Layout Assignment**

INPUTS: The unassigned area to be inserted.

OUTPUTS: The layout assignment is redrawn to include the area and the associated cost updated.

DESCRIPTION: With this command, an unassigned area is inserted into a layout assignment. If the merged assignment violates a layout constraint, the planner is prompted to confirm this is acceptable.

FUNCTION: **Delete a Area From a Layout Assignment**

INPUTS: Area to be deleted.

OUTPUTS: Layout assignment redrawn without specified area.

DESCRIPTION: With this command, an assigned area is deleted from the layout assignment to which it is currently assigned. Once this command is selected, then the area to be deleted is specified. The system will free the demand area.

FUNCTION: **Merge Two Blocks**

INPUTS: Two blocks.

OUTPUTS: The two block assignments are erased and the combined block assignment is redrawn. The associated cost is updated.

DESCRIPTION: With this command, the Areas in two layout assignments are combined to form one assignment. The layout skeleton of the first assignment is taken to be the layout skeleton representing the combined layout. The planner must select this command, specify the first layout, then specify the second layout. The system will combine both facilities and draw a combined layout assignment. If the merged layout assignment violates a layout constraint, the planner is prompted to confirm if this is acceptable.

FUNCTION: **Separate a Block Into Two Blocks**

INPUTS: The block to be separated and the areas to be assigned to each new block.

OUTPUTS: The original block is erased and the new block assignments are redrawn.

DESCRIPTION: To separate a block into two blocks the planner must first select this command, then specify the area to be included in each area of the new layout. This is accomplished by answering a yes/no question which the system presents for each area in the current block assignment. Before each question is asked, the system places the cursor on the particular area. The system will create a new block made up of the indicated areas. The system will make two copies of the original layout skeleton - one for each of the new blocks.

FUNCTION: **Clear All the Areas in a Layout Assignment**

INPUTS: Layout.

OUTPUTS: All areas in the layout assignment are freed.

DESCRIPTION: When the planner selects this command, the areas assigned to a current layout are cleared (re-specified as unassigned). This command is selected and the planner specifies the layout to be cleared. The layout assignment is erased but the skeleton representing the layout remains.

FUNCTION: **Add Areas to Layout**

INPUTS: Number of areas to be added.

OUTPUTS: As areas are added, the layout assignments are redrawn.

DESCRIPTION: This command sequentially adds unit grid areas to the current blocks. On selecting this command the planner is prompted for the number of areas to be added. The system will then sequentially add areas to the current solution until the specified number of areas have been added, or until no further additions are possible with the current set of layout skeletons. An entry of **0** (zero) will cause no areas to be added.

FUNCTION: **Block Interchange**

INPUTS: Blocks to be interchanged.

OUTPUTS: Layout with blocks interchanged.

DESCRIPTION: This command interchanges the positions of two blocks in the layout. After selecting this command the planner is prompted for the blocks to be interchanged. If the blocks are the same size, the system will interchange them. If the blocks are not the same size, the system prompts for how the interchange is to be made.

FUNCTION: **Draw Aisle Structure**

INPUTS: Block layout.

OUTPUTS: Block layout with aisle structure and associated material movement cost and aisle area.

DESCRIPTION: This function allows the planner to configure the aisle structure on the block layout. If the aisles violate any constraints on the layout, the systems prompts the planner to confirm the positioning of the aisles.

FUNCTION: **Position Input/Output Stations**

INPUTS: Block layout and number of stations for each block.

OUTPUTS: Block layout with stations located.

DESCRIPTION: This function optimally locates the input/output stations within a block with respect to minimizing travel costs.

FUNCTION: **Position Icons**

INPUTS: Blocks to be included in the layout

OUTPUTS: Block layout with icons (chairs, machines, etc.) drawn on the layout.

DESCRIPTION: This function allows the user to call from memory icons representing specific equipment and position the equipment icons in a detailed layout of the block.

Algorithmic Effort

As can be seen from the descriptions of functions in this section, the implementation of the workstation will require coding of a large number of algorithms. All of these models and algorithms have been tested in a research environment with very positive indicated potential. Some of them are currently in common use in logistics planning. The availability of so many different procedures in a user-friendly environment is the real power of the logistics workstation.

VI. PROTOTYPE

A prototype of the workstation has been developed to demonstrate the concept feasibility and indicate its value in solving logistics planning problems. The prototype also illustrates that existing microcomputer technology is adequate to handle the advanced graphics and computational requirements of the workstation. The problem area selected for the prototype includes basic vehicle routing and warehouse location.

Although they are clearly related, problems of warehouse location and vehicle routing have traditionally not been solved within a common framework. Warehouse location problems are generally addressed using linear functions of distance between customers and the warehouse as cost measure. This may not be a reasonable approximation of cost if deliveries to customers are not in whole truck loads. For this case there are typically multiple deliveries on a route. The prototype provides for these problems to be linked either sequentially by first solving a location problem and then generating appropriate routes for each warehouse, or simultaneously by building routes as customers are assigned to a warehouse. Hence, in addition to demonstrating utility of the workstation with regard to the separate problems of warehouse location and vehicle routing, the prototype demonstrates feasibility of considering these two problems together.

Of the four workstation modes, only analysis mode is included in the current prototype. The other modes require substantial programming effort, but the design provided in this report is well within the capability of current personal computer hardware and operating systems.

Files

A commercial editor external to the prototype is used to create and edit files. For the prototype, the following file handling functions have been developed.

Read a file into the workspace

Save a file form the workspace

Look at contents of a file

Discard a file

Display

Since both routing and location problems have a geographical representation, most of the display functions in the prototype allow geographical display of data. The prototype display functions are listed below. The concept of a skeleton will be discussed in the analysis section of this chapter.

Plot customer locations

Plot warehouse locations

Draw routes

Draw background map

Draw time distance chart

Identify a customer

Identify a warehouse

Assign color to customer cluster

Zoom an area

Restore (unzoom) the original display window

Draw route skeletons

Draw facility skeletons

Redraw a current graphics screen

Some of the display capability of the workstation is illustrated in Figure 1. The menu items in green indicate some of the display functions available. The small points in the figure represent customer locations and the larger point represents a warehouse. The points are drawn to a geographic scale.

Reporting

To demonstrate reporting capability required, four basic reports are generated in the prototype: Route summary report, Individual route report, Customer report, and Location report. The workstation design includes a generalized reporter allowing the user to customize reports for particular applications.

A route summary report is illustrated in Figure 2. This is one of the reports available in the prototype.

Analysis

To demonstrate power of the workstation concept, a variety of analysis tools are included in the prototype. The concept of generating design skeletons as the basis for the planning process has proven extremely effective in aiding logistics planners. For the prototype, skeleton generation is available for routes and warehouse locations. For routes, both lines and points are allowed as skeletons. The basic idea is that in adding new points to routes, rather than try to compute an insertion cost based on the actual route, cost is based on the skeleton. For example, if the skeleton is a point, then the cost of inserting a new customer into the skeleton is taken to be the cost from the customer to the skeleton point and back. Two things are accomplished with a skeleton. First, calculations are much faster since the actual route is more complex than the skeleton. Second, the user is able to guide the shape of routes through location and structure of skeletons.

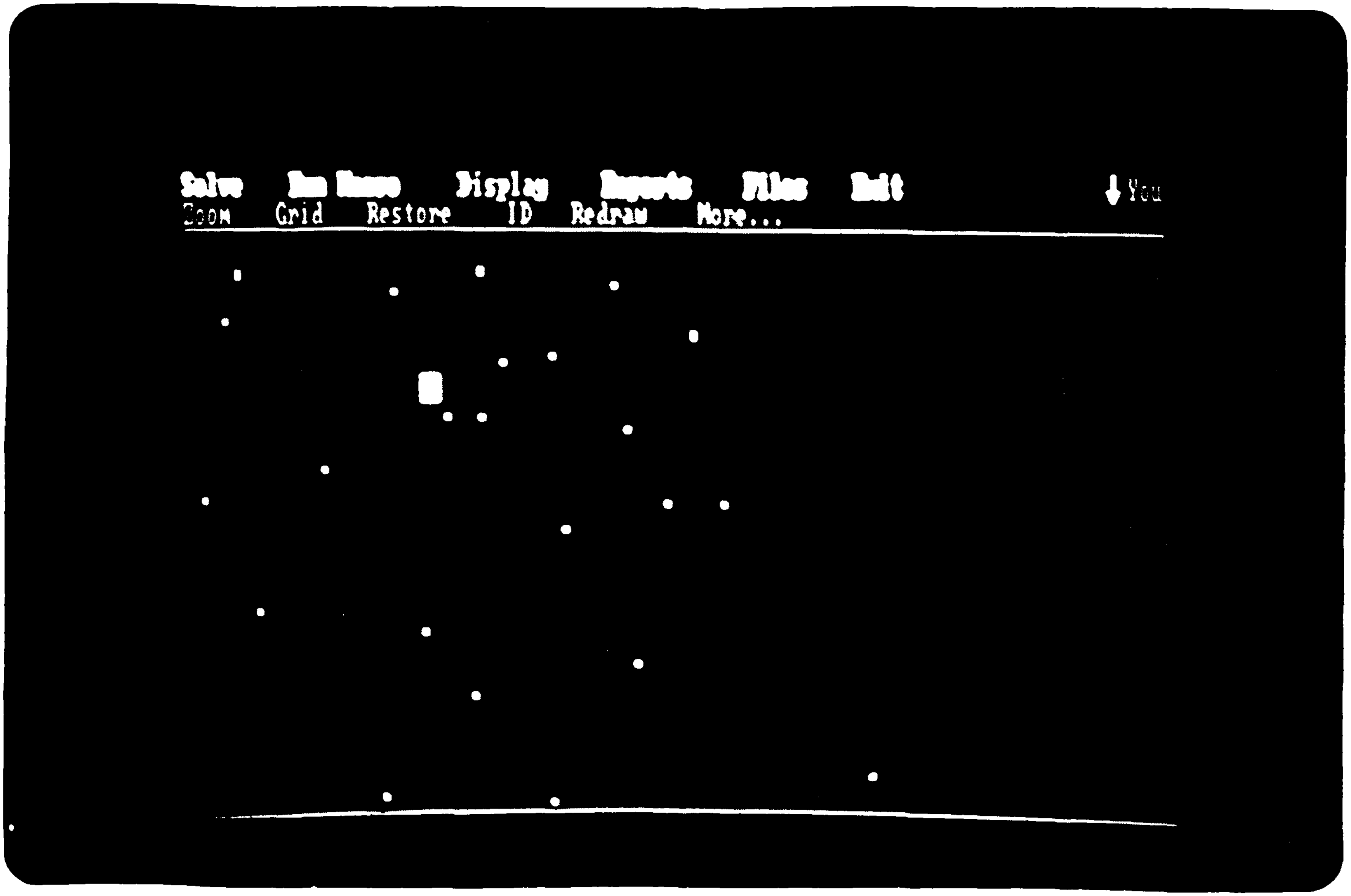


Figure 1. Illustration of Workstation Display Functions (Plot Customer Locations and Plot Warehouse Locations)

```
                         TOTAL ROUTES SUMMARY
Problem    SAMPLE                   No. Routes     7       Dist/Grid  0.200
Solution   SAMPLE             Dist/Time Zone 1   1.000     Zone 2     1.000
Map Name   NOMAP                        Zone 3   1.000     Zone 4     1.000
        Pickup          Delivery
Rte   Vol %   Wgt %   Vol %   Wgt %    Time %  #Pts Depot    Vehicle      $
 1     47     100      0       0       163  27   4  CULLMAN  VEHICLE1   104
 2     44      90      0       0       178  30   2  CULLMAN  VEHICLE1   113
 3     34      90      0       0       381  63   5  CULLMAN  VEHICLE1   243
 4     43      95      0       0       366  61   5  CULLMAN  VEHICLE1   233
 5     36      87      0       0       218  36   3  CULLMAN  VEHICLE1   139
 6     24      95      0       0       142  24   3  CULLMAN  VEHICLE1    90
 7     12      20      0       0       123  20   1  CULLMAN  VEHICLE1    78



Routed  Vol      1930     Unrouted  Vol        0       Total Cost  $1000
        Wgt     23100     (window)  Wgt        0             Time   1569.46
        Points     23               Points     0             Points 23

SUMMARY 1=Back 2=Fwd 3=Total 4=Route 5=Cust 6=Report 10=Return
```

Figure 2. Illustration of Route Summary Report

A variety of analysis functions are included to specify, position, and optimally locate skeletons for route and warehouse locations. Only points are used for warehouse locations. There are also functions to aid in assigning points to routes, both manually and based on optimization concepts. Analysis functions included in the prototype are indicated below.

Insert a customer into a route

Delete a customer from a route

Merge two routes

Separate a route into two routes

spacefilling curve tour

2-opt

Assign a customer to a facility

Delete a customer from a facility

Merge two facilities

Separate a facility into two facilities

Assign a facility to a site

Delete a facility from a site

Add a route skeleton

Delete a route skeleton

Position a route skeleton

Add a facility skeleton

Delete a facility skeleton

Position a facility skeleton

Macros

The large number of functions which a user might wish to use make the concept of macros very attractive. If the end user is someone with very limited knowledge of optimi-

zation methodology, the following process might be an attractive one to use in generating a set of delivery routes. (a) Generate a set of route skeletons. (b) Assign all customers to route skeletons. (c) Route all customers assigned to skeletons. (d) Optimally reposition the route skeletons. (e) Free all customers from routes. (f) Repeat steps (b)-(e) until there is no improvement in routes.

A similar strategy for locating a set of warehouses would be as follows. (a') Generate a set of warehouse location skeletons. (b') Assign all customers to location skeletons. (c') Optimally reposition the location skeletons with respect to the customer assignment. (d') Reassign customers with respect to the new skeleton locations. (e') Assign the location skeletons to potential warehouse sites. (f') Reassign the customers with respect to the new skeleton locations. (g') Repeat steps (b')-(f') until there is no improvement in locations.

Macros to perform this set of steps for both vehicle routing and warehouse location are included in the prototype workstation. If desired the planner could generate routes for warehouse locations by altering the sequence in which these macros are executed. Macros included in the prototype to date are listed below.

Free all customers from routes

Free all customer from a facility assignment

Assign all customers to routes

Assign all customers to facilities

Assign all facilities to sites

Initiate new route skeletons

Initiate new location skeletons

Resequence all routes

Position route skeletons

Position location skeletons

The results of two of the macros are illustrated in Figures 3 and 4. Figure 3 indicates the routing of vehicles among a set of deliveries from a single warehouse. The routes are color coded to correspond to the route summary report.

Figure 4 indicates the location of six warehouses (each represented by a large X) and the corresponding customers to be served by each warehouse. The customers are color coded corresponding to the warehouse which will serve them.

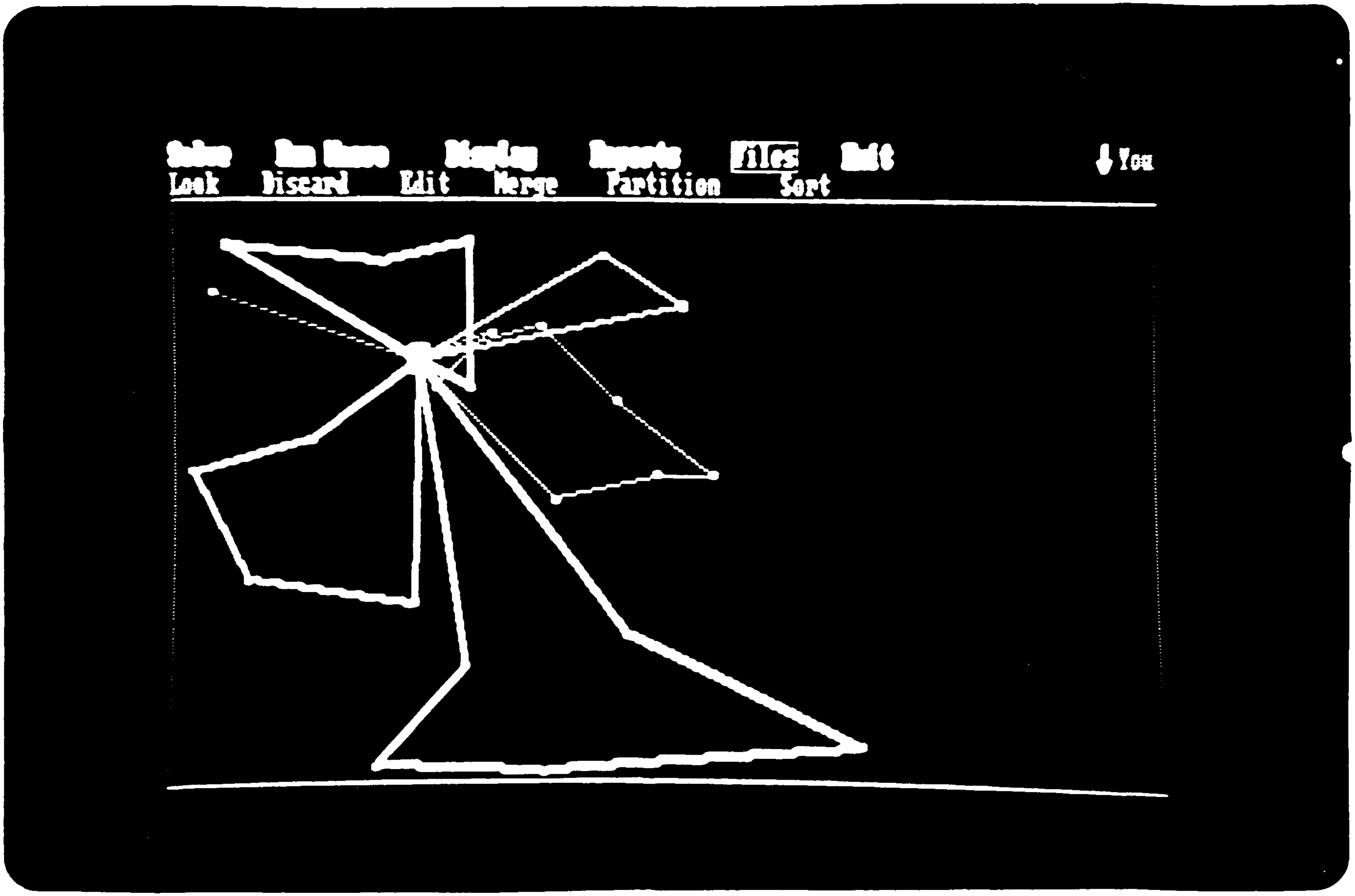


Figure 3. Illustration of Workstation Macro Which Assigns Customers to Routes and Sequences Routes

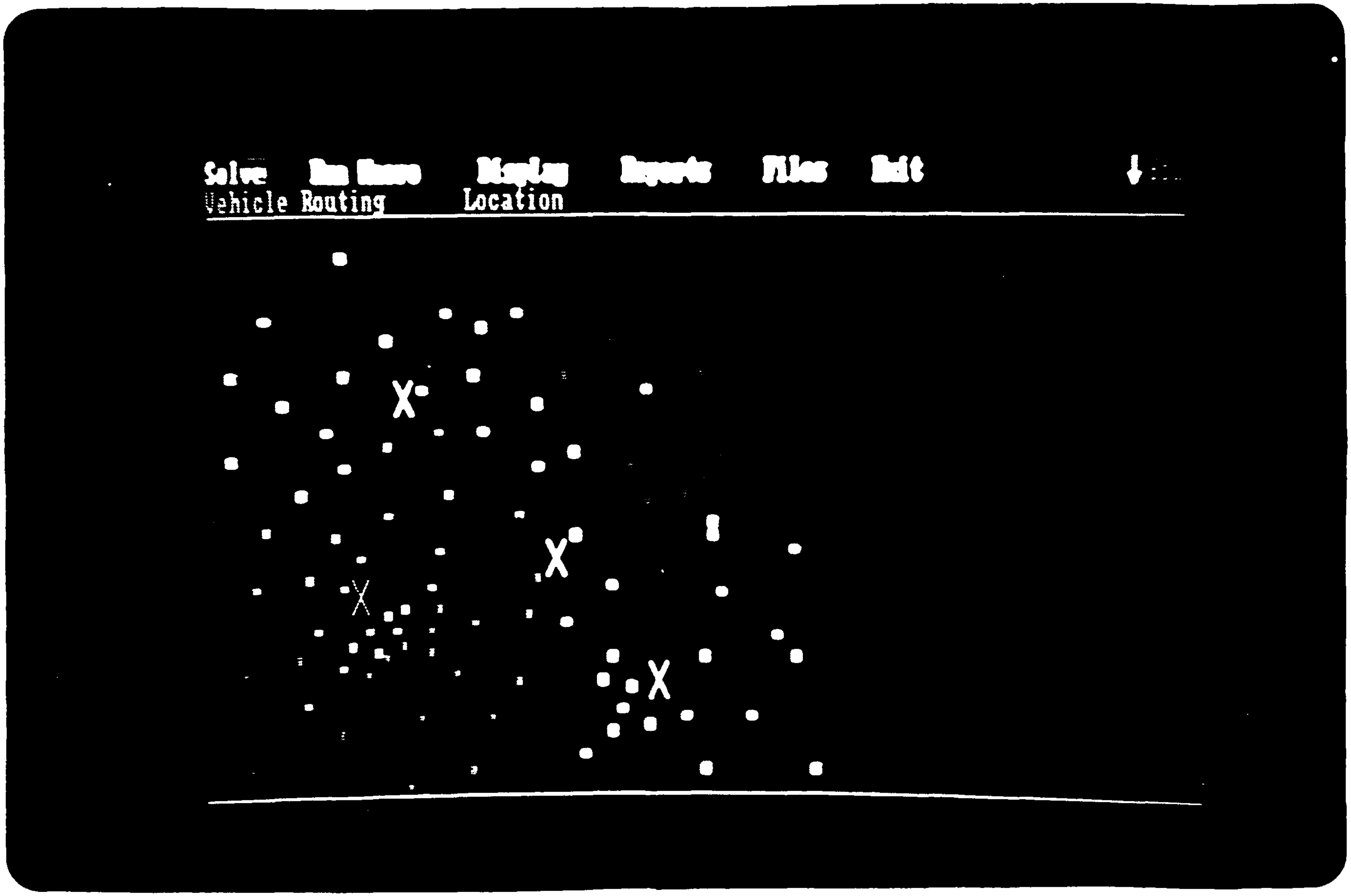


Figure 4. Illustration of Workstation Macro Which Locates Warehouses and Assigns Customers to Warehouses

VII. IMPLEMENTATION

Hardware

The workstation is designed for use on the popular IBM PC (and compatible) family of microcomputers. This class of microcomputer has emerged as an industry standard with over two million machines in use, and it enjoys a large and growing base of software and hardware support. The IBM PC has the computing power and graphics support needed for the workstation. Most important, the industry is committed to insuring that future PC performance improvements are compatible with existing technology. These factors make the IBM PC the logical choice for the workstation.

Because graphics are a fundamental component of the workstation, the workstation PC will require a graphics monitor and adapter board. High-quality graphics systems such as IBM's Enhanced Graphics System will provide maximum workstation benefit. A mouse system is recommended for easy user-computer communication. A math coprocessor chip is recommended to perform the complex calculations that many of the system functions require. Fully populated system memory is needed to handle the size of the workstation software. All of these options are inexpensive and readily available.

Software

Current software available for the IBM PC is mature, drawing on several years experience in PC software design techniques. The workstation will employ the best of these techniques, utilizing proven menu control structures, macro building procedures, and on-line help commands.

The workstation will primarily be coded in the C programming language. C is a high-level modular language which enjoys wide popularity in scientific applications because of its

speed, power, and portability. A wide range of C language support utilities for the PC are available to handle some of the complex workstation operations.

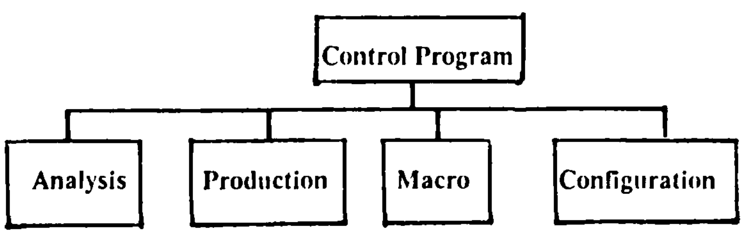
An important feature of the C language is its library management capabilities. C permits the user to create personal libraries of software routines. This feature will allow the workstation to keep up with evolving logistics optimization techniques. The workstation user can write his own modules and add them to the workstation system function library.

The limited amount of PC memory available coupled with the size of the workstation software requires the usage of overlays. This technique keeps most of the software on disk rather than in memory. Software modules are brought into memory only as needed by the workstation. For example, a workstation running in Analysis mode would only keep in memory controlling programs and the default set of Analysis mode programs. The software needed to run the other three modes are not needed and thus kept on disk.

The workstation will use the Virtual Device Interface (VDI) graphics system, a microcomputer graphics standard used by IBM. VDI allows the software developer to write graphics software independent of any particular hardware device. Device drivers external to the application adapt this graphics software to a particular hardware device. This independence of hardware devices is important in the rapidly changing computer industry.

Main Control Structure

Primary workstation operations are directed by the **Control Program**, or **CP**. The CP is the workstation office manager, controlling the operation of Analysis, Production, Macro, and Configuration modes. The main control hierarchy is:



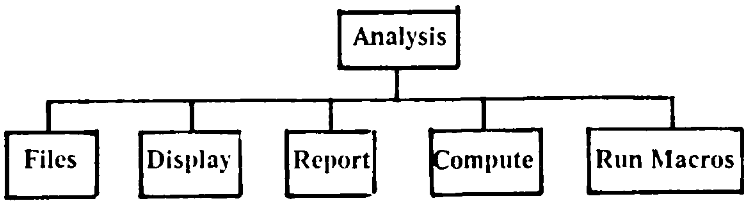
The CP is permanently resident in memory when the workstation is used. The CP controls workstation operations that are independent of mode. These operations include initialization and management of the physical environment (such as the graphics display, mouse, keyboard, and printer), allocation of memory for mode operation, and system error handling procedures.

Help

The on-line help module deserves special mention here because it applies to all operations of the workstation. Although the workstation will be easy to use, the scope of the workstation is large enough to require a complete and easy-to-access help module. To get help on any workstation command the user first presses a function key reserved exclusively for help. Then the workstation command for which help is desired is executed. Help screens detailing the purpose and usage of this command are then displayed.

Analysis Mode Control Structure

Analysis mode contains the basic tools for interactive problem development and analysis within the workstation. The Analysis mode control hierarchy is:



Files are the basic input and output structure for many of the workstation system functions. For example, the system function **Draw Points** requires an input file of points and their locations, such as customers and their (x,y) coordinates. The system function **Spacefilling Curve Tour** requires an input file of points and their locations, and generates an output file indicating the order the points are visited.

Files allow communication between system functions. For example, the output file of the **Spacefilling Curve Tour** system function could be the input file to the **Break Up Great Route Into Skeletons** system function. This system function generates an output file of skeletons, which could be an input file to system function **Add Points**. In this manner a vehicle routing problem could be solved by running sequences of system functions.

Files are comprised of records and fields. A record is a row entry in the file, and fields are the column entries of the row. For example, a file could be:

```
COMMENT: military airlift
C141    OVERSIZED    30.1    425    2000
C5      OUTSIZED     70.9    445    3200
```

The first record is an optional comment line. Comment lines can be spaced throughout the file, and are used to describe the file. Comment lines begin with the reserved word "COMMENT:". This file has two records of military airlift planes. The first field is the name, the second field is the cargo class, the third field is the tonnage constraint, the fourth field is the air speed, and the fifth field is the flying range.

Workstation files cannot contain records of different types. For example, a file of warehouse records and vehicle records cannot be used in the workstation. Also, all workstation data files must end with the file extension ".LWD".

In addition to specifying input and output files when using system functions, the user must also specify file **Field Maps**. The field map is a file that tells the system function the roadmap of a corresponding input or output file. Specifically, it indicates the size of the fields, and what information is in the fields. A field map could look like this:

```
  FILE: DEMANDS
  NAME: 1,10
     X: 11,20
     Y: 22,30
WEIGHT: 32,40
```

This is a field map of the file DEMANDS.LWD. It indicates that the record name is in columns 1 to 10, the x coordinate is in columns 12 to 20, and so on.

There are many reasons for using field maps instead of fixed field locations in files. First, there may be more than one value for a certain parameter. For example, a plane may have normal and overload capacities and flying speeds, or a customer may have several different service time windows, or the user may wish to use latitude and longitude in place of an (x,y) coordinate pair, or a transportation network moving volume may be solved in place of one moving weight. Different field maps of the same data file let the user easily swap input parameters to a system function.

Second, the field map specifies to the system function the active input parameters. For example, the user can locate a new warehouse with respect to a number of criteria. A field map specifying only customer coordinates tells the Position system function to locate the warehouse based on centroid. A field map specifying customer coordinates and weights tells the **Position** system function to locate the warehouse based on weighted centroid.

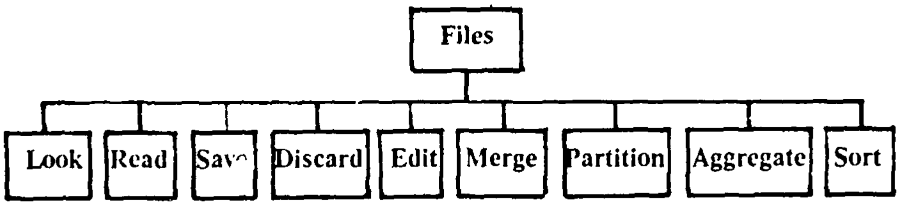
Third, field maps allow the user to keep all relevant information about objects in a master data file, rather than many files for use with different system functions.

All field map files must end with the file extension ".LWF". Also, each workstation data file has a default field map file associated with it. The default field map file name is the

same as the data file name. For example, the data file "DEMANDS.LWD" has a default field map "DEMANDS.LWF". This feature is useful because a data file will likely have a single field map which never changes. When executing a system function the specification of field maps is optional. If no field map is specified, the default field map will be used. The user can always override the default field map by specifying a different field map.

The proper use of data files and field map files is critical to the workstation. The workstation is helpless if the correct logistics problem cannot be communicated to it. For this reason the workstation will possess extensive data file maintenance routines. One module will display a data file overlaid with a field map. The overlay will be color-coded so the user can instantly determine if the field map is defining the correct fields of the records. Another module will allow the user to view input data received by a system function record-by-record, field-by-field.

The file control hierarchy is:



The **Files** command brings up a full-screen catalog of existing workstation files. The user navigates a high-lighted bar through the file catalog using the arrow keys on the PC keyboard. When the bar is high-lighting the desired file the user issues the selected sub-command. **Look** allows the user to view the contents of a file. **Read** allows the planner to place a file into the workspace for internal processing. **Save** saves files from the workspace to disk. **Discard** is used to delete files from disk. **Edit** will call a commercial editor to allow changes to the files. This editor is chosen and supplied by the user. The editor will either be chained to the workstation through a parent-child process, or will be accessed through

batch level commands. **Merge** allows two or more files to be combined, while **Partition** allows files to be decomposed into pieces. **Aggregate** takes a file as input and creates an aggregated output file. The type of aggregation is chosen from a supplied menu, and is based on a field or fields in the file. For example, the user may wish to aggregate aii customers within a Euclidean distance of 1 mile, or all ships with a speed between 20 and 25 knots.

Display contains the set of system functions which control the graphical display of information. The entire display system function will not fit in memory simultaneously. The user selects logical groupings of this library to be resident in memory. For example, the user may bring in those **Display** system functions which are used in solving a vehicle routing problem.

Report can generate reports from files. The user selects a default report format, and then has the option to customize this format. For example, the route summary report format can be selected, and customized to show only routes in a certain customer delivery zone.

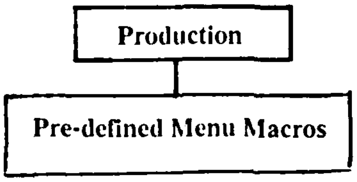
Report is a comprehensive and powerful reporting system for files. A library of common report requests is supplied to the user. The user can select one of these for execution, modify it to better fit his needs, or construct entirely new requests to be added to the library. Each report request consists of directives for the appearance of the desired output with respect to what fields are to be printed, and where the fields are to appear on a printed page. A flexible but powerful method for selecting only certain records of the file is provided in the context of an information algebra-based retrieval language based on a data dictionary formed by the combination of the generic labels defined by the field map files and optional supplemental data dictionary definitions specific to particular files supplied by the user. This generalized reporting capabilities enables the user to specify reports on files detailing which records are selected for reporting, what is to be reported for each record, and the sequence of records appearing on the report.

Compute contains the set of system functions that perform optimization or computation tasks. Like the **Display** module, only a subset of the **Compute** system function library can fit in memory at one time. The user selects logical groupings of this library to be resident in memory.

Run Macros allows the user to execute pre-defined macros. Pre-defined macros are stored in files and thus can be accessed in Analysis mode.

Production Control Structure

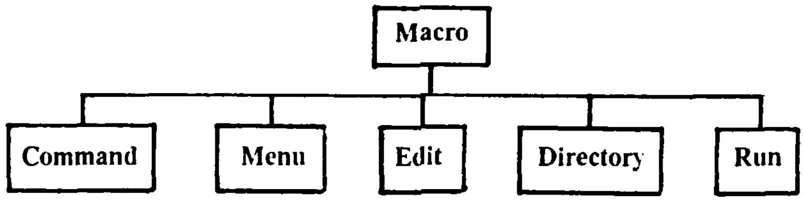
Production mode uses well-defined systems to solve a particular class of problem. These systems are assembled using the other three modes. The Production mode control hierarchy is:



The control structure is restricted to pre-defined **Menu Macros**. These macros access pre-defined **Command Macros**.

Macro Control Structure

Macro mode allows the user to create macros containing sequences of system functions, or definitions of menu structures. The Macro control hierarchy is:



Command macros and **Menu macros** can be built in Macro mode. Command Macros contain sequences of system functions, and can be constructed by **Command** or **Edit**. **Command** allows the user to simulate the execution of system functions in the same manner the functions are used in Analysis mode; the system functions are not executed, but rather "captured" into a macro which preserves their order and input/output arguments. The same control structure used in Analysis mode to access **Display** and **Solve** system functions is used in the **Command** module.

Edit brings up a file maintenance module similar to the **Files** module in Analysis mode. Macro files are accessed via full-screen arrow manuevers, and sub-commands allow deleting, merging, partitioning, or editing of the files. Editing is performed by a commercial editor in similar to the **Files** module editing procedure.

A Command Macro is stored in a file with file extension ".CMC". A Command Macro can call any workstation system function, and call other Command Macros (the calling name of a Command Macro is its file name without the file extension). A Command Macro could look like this:

```
COMMENT: This macro generates a travelling salesman tour
COMMENT: through a set of customers in the data file
COMMENT: DEMANDS.DAT. The tour is stored in the data
COMMENT: file TOUR.DAT. The tour is then improved using
COMMENT: 2-opt and limited 3-opt techniques until no
COMMENT: more improvements can be made.

COMMENT: solve TSP
        TSP(DEMANDS,TOUR);

COMMENT: initialize variables which will indicate how
COMMENT: many swaps the 2-opt and limited 3-opt performed
        NUM_2_SWAPS = 1;
        NUM_3_SWAPS = 1;

COMMENT: begin 2-opt and limited 3-opt until each
COMMENT: made no swaps
        WHILE( (NUM_2_SWAPS > 0) AND (NUM_3_SWAPS > 0) ) {
            2_OPT(DEMANDS,TOUR,NUM_2_SWAPS);
            LIM3_OPT(DEMANDS,TOUR,NUM_3_SWAPS);
        }
```

Command Macros use their own macro programming language. Although not nearly near as complex as a programming language such as C, the Command Macro language will allow variable names and logical control, and have defined reserved words such as "WHILE" in the macro above. This language will closely resemble traditional programming "pseudo-code".

The processing of the macro programming language is performed in two parts: "parsing" and "syntactical analysis". The processes are used by Analysis, Production, and Macro Modes and are thus a part of the CP.

The parser takes a macro command line as input and outputs a data structure called a "uniform symbol table" (UST). The UST consists of several entries, one for each language element discovered in the command macro line. There are several types of language elements. These include:

1. **Keywords** - macro language reserved words such as COMMENT, WHILE, and AND.

2. **Data Dictionary Names** - items which are defined as generic field map identifiers or are associated with a file using a user-specified relationship.

3. **Relational Operators** - comparison operators such as >, <, =, and NOT.

4. **Arithmetic Operators** - +, -, *, /, etc.

5. **List Operators** - min, max, etc.

6. **Mask Operators** - such as LEFT("LITERAL",3) = "LIT".

7. **Literal Delimiters** - double and single quotes.

8. **Separators** - : or , or ;.

9. **Associators** - parentheses and braces.

10. **Logical Conjunctions** - AND, OR.

11. **Literals** - constants (numeric and alphanumeric).

Each entry of the UST has a number of fields. Some of these are essential for the interpretation of the macro command during the syntactical analysis process, while others aid in the diagnosis of incorrect input. Some of the fields of a UST entry are:

1. Type of language element (KEYWORD, SEPARATOR, etc.)
2. Actual language element (WHILE, AND, etc.).
3. Beginning column of element on input statement.

There are several other fields for the UST entry not described here.

For example, the macro command line

WHILE (NUM_2_SWAPS + 2 > 13

would generate the following UST:

<u>Type</u>	<u>Element</u>	<u>Beginning Column</u>
KEYWORD	WHILE	1
ASSOCIATOR	(	6
DATA DICT NM	NUM_2_SWAPS	7
ARITH OPER	+	18
LITERAL	2	19
RELATIONAL OPER	>	20
LITERAL	13	21
ASSOCIATOR	>	23

Once a macro command line has been parsed, a tree-based syntactical analysis procedure translates the UST into the actual execution of the specified macro command, much like a language interpreter (such as BASICA on the IBM/PC) operates on an input source language and performs the desired functions.

Menu contains utility functions to aid in constructing menu macros. These Menu Macros are used in Production or Analysis mode, and are stored in files with file extension ".MMC". The format of these files is:

name : macro name : description
name : macro name : description
name : macro name : description
etc.

For example, a Menu Macro could be:

Auto_route:	ROUTE:	solve a routing problem
Display:	DISPLAY:	display system functions
Short_path:	SHORT PATH:	run shortest path algorithm

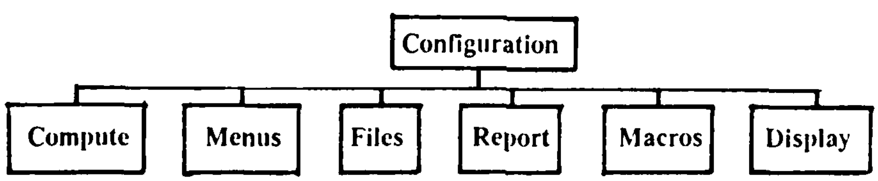
In this example the first macro is a command macro solving a vehicle routing problem. The second macro is a menu macro which defines the menu list to be displayed if this macro is selected. The last macro is a system function. If SHORT PATH were selected, the system function which solves a shortest path problem would be executed. Macros in a Menu Macro file may be either Menu Macros, Command Macros, or system functions.

The name of a Menu Macro is its file name (without the extension). The descriptions are used in a manner similar to the Lotus 1-2-3 convention. The description of a macro appears in the menu when the menu cursor is placed over that macro's name.

The workstation will have substantial utilities to aid the user in constructing menus for a Production mode system. One utility will draw a menu decision tree. The user specifies a Menu Macro, and all macros and system functions called by this macro are shown, as well as any lower depth calls of macros and system functions. Error checking is continually performed by the CP in Menu Macro construction to insure that all macros are exist and contain valid instructions. If a certain Menu Macro contains too many options to be displayed on one screen, the CP will manage paging of the options.

Configuration Control Structure

Configuration mode is used to specify controlling parameters for the workstation. This mode allows the user to define the system functions available in Analysis mode and the menu structures that access them. The Configuration control hierarchy is:



Configuration mode is system bookkeeping module which simply manages the active system function modules, menu structures, macros, and report modules which are available to Analysis mode.

VIII. SUMMARY AND RECOMMENDATIONS

This report delineates a logistics workstation design which will provide planners with state-of-the-art modeling and analysis capability in a user friendly interactive environment. The workstation will provide a powerful tool to overcome the burdens of developing models and procedures, structuring data, and formatting results so that the user can analyze logistics problems and design logistics systems. The prototype developed demonstrates both the feasibility of the concept and the value of an interactive workstation.

Summary

The workstation provides a framework for integrating advancements in data base management, interactive optimization, and computer graphics to provide a comprehensive logistics problem solving environment. The flexibility of the files management subsystem will enable logistics planners to conveniently work with the same data in a variety of different ways to address different problems. The user will be able to apply the appropriate methodologies to aid in understanding and analyzing specific problems. The breadth and flexibility of the workstation optimization functions provides a set of extremely powerful analysis tools. With these tools, logistics planners will be able to address complex problems without the prerequisite mathematical expertise previously required for sophisticated analysis. The graphics functions provide significant insight into both problem structures and potential solutions. This provides the user the capability to realistically evaluate appropriate models and work interactively with them in solving broad classes of logistics problems.

Where possible, the logistics workstation design incorporates available software. For example, the design includes a capability for the planner to utilize available text editors in creating and modifying files. The workstation is able to accomplish this by proper chaining,

memory management and control. It will also be able to interface with computer aided design packages to facilitate layout preparation.

A prototype has been developed which demonstrates both the feasibility and the potential value of the workstation. The prototype addresses basic vehicle routing and warehouse location. It contains fundamental functions for data manipulation, graphical data display, optimization, and reporting.

Building on the knowledge gained from the prototype, the report describes a design for a more general logistics workstation. This workstation ~~will be able to~~ operate in different modes (production, analysis, etc.) depending on the sophistication of the user and the desired analysis. This will allow planners to use the workstation for quick analyses, but will also enable them to develop systems which can be utilized by planners who are functionally oriented rather than analysis oriented.

The report includes the necessary modeling and control structures for the logistics workstation. It also describes a variety of functions to be included in the workstation design, together with their inputs and outputs.

The technology detailed for the workstation, including the optimization and computational algorithms, is all currently available. However, most of the technology will have to be specially coded to integrate into the workstation framework.

The technology selected for the workstation is powerful and efficient. For instance, the algorithms selected for the routing functions have been tested on the major benchmarks for such problems; and they have produced the best known solutions in every instance, and in one case they found a better solution. In the prototype, similar algorithms are able to solve 1500 point problems in a reasonable amount of time using an IBM PC AT. Also the network algorithms perform at state-of-the-art levels, solving very large problems on a micro-computer in a short amount of time.

The workstation design is reasonably independent of the implementation hardware. This will allow the workstation to be somewhat portable among alternative computing environments and to take advantage of evolving technologies in hardware and software.

Recommendations

It is recommended that the initial workstation implementation address the four broad problem families of logistics families discussed in section II: vehicle routing, facility location, warehousing, and facility layout. Many of the problems associated with these families have common data structures, graphical requirements, and optimization methodologies. In spite of their common elements, these problem families cover a very broad range of logistics problems. Hence, they are logical problem grouping to use for the initial workstation implementation.

Successfully fielding the workstation requires several important steps. First, a detailed system specification must be created which contains the variable definitions, subroutines, and necessary linkages to complete the coding. Second, the components (subroutines, etc.) must be coded in accordance with the specification. User documentation must be developed to provide complete functionality, use, and application of the workstation. Finally, thorough quality assurance and testing must take place on the system and documentation to ensure that the highest quality product is fielded.

The code generated for the prototype can be directed integrated into the general system. The functions already developed for the prototype will allow testing of the various control elements of the workstation as they are developed.

The design proposed here when implemented will provide the most powerful tool available for addressing logistics problems associated with routing, location, warehousing, and layout. It will also provide the basis for addressing a variety of logistics issues outside these families since both the optimization models and the data handling capability apply to a broad class of decision problems.

END

DTIC

7-86